Information-Oriented and Energy-Aware Path Planning for Multiple Air-Dropped Small Unmanned Aerial Vehicles

José Bento
Institute for Systems and Robotics, Instituto Superior Técnico,
Lisbon, Portugal
0009-0005-3608-3391

Meysam Basiri
Institute for Systems and Robotics, Instituto Superior Técnico,
Lisbon, Portugal
0000-0002-8456-6284

Rodrigo Ventura
Institute for Systems and Robotics, Instituto Superior Técnico,
Lisbon, Portugal
0000-0002-5655-9562

2024 7th Iberian Robotics Conference (ROBOT) | 979-8-3503-7636-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ROBOT61475.2024.10796923

***Abstract*—The use of small Unmanned Aerial Vehicles (UAVs) has revolutionized operations such as Search and Rescue (SAR) or surveillance and monitoring. The problem remains of how to best utilize these assets given their limited endurance while taking advantage of prior target position information. The goal of this paper is to maximize the likelihood of quickly locating targets by utilizing a novel formulation of the Coverage Path Planning (CPP) problem, and applying a global optimization algorithm. The approach described takes advantage of the availability of multiple UAVs that can be air-dropped anywhere in the Area of Interest (AOI). This is particularly useful for scenarios where the AOI cannot be completely covered by a single UAV due to its limited endurance, and where the expected target location is represented by a Probability of Containment (POC) map. By prioritizing areas with a high POC, the algorithm increases the probability of target detection and reduces detection time, thereby enhancing the likelihood of survival in SAR missions. The presented algorithm outperforms the baseline Boustrophedon algorithm typically used for area coverage.**

***Index Terms*—Multi-UAV path planning, Search and Rescue, Energy-aware, Information-oriented, Aerial deployment**

I. Introduction

The affordability and availability of small Unmanned Aerial Vehicles (UAVs) makes them suitable for use in large numbers for a wide range of applications. In this paper, the use of an air-dropped swarm of small UAVs in Search and Rescue (SAR) missions is explored. Utilizing these vehicles eliminates the need for larger platforms, significantly reducing the cost and the risk to human assets. Additionally, low-flying UAVs can be particularly advantageous in situations where the visibility from higher altitudes is reduced due to atmospheric conditions such as fog or smoke. Although this paper focuses on SAR missions, the approaches presented are adaptable to other applications such as Surveillance and monitoring.

The main contribution of this paper is a novel multi-UAV path planning algorithm for a SAR scenario that maximizes the likelihood of finding the target while minimizing the time it takes to do so by optimizing the deployment point and path of the search vehicles. The approach presented is information-oriented, leveraging prior information regarding the expected target position, as well as energy-aware, by considering flight time limitations of the search vehicles. It also takes advantage of the fact that the UAVs are air-dropped and therefore have a flexible deployment position.

We first decompose the Area of Interest (AOI) into a grid of small cells, with each cell assigned a value representing the likelihood of the target's presence, thus generating a Probability of Containment (POC) map. Subsequently, a multi-robot discrete path planning algorithm with variable initial positions is applied to find a path that maximizes an objective function aligned with the mission goals.

This paper builds upon the authors' previous work [1], which proposed a Simulated Annealing-based algorithm for single-UAV path planning with a fixed deployment position, and extends it to multi-UAV operations. This paper is structured as follows: in section II a literature review is performed on the state-of-the-art methods for Coverage Path Planning (CPP) problems, energy consumption studies of small UAVs, and strategies that leverage prior information. In section III the problem is defined and some metrics for the mission success are presented, along with an objective function. In section IV, two approaches for maximizing the objective function are formulated, the results of which are presented in section V, along with some experiments to tune the parameters. Finally, section VI contains some final remarks and outlines possible future work.

II. Related works

The problem of finding a target in a given area can be approached as a CPP problem, where an AOI is defined that is intended to be fully covered by the search vehicle. The author in [2] proposes a classification for CPP problems based on their decomposition of the AOI, coverage guarantees and the available knowledge of the environment, and describes three main methods for decomposing the AOI. Exact cellular decomposition, used in [3], where the AOI is divided into smaller trapezoid-shaped subareas that are individually

This work was supported by Aero.Next project (PRR - C645727867-00000066) and LARSyS funding (DOI: 10.54499/LA/P/0083/2020, 10.54499/UIDP/50009/2020 and 10.54499/UIDB/50009/2020).

covered by performing back-and-forth motions inside each one, and moving between adjacent subareas until the whole AOI is covered. Approximate cellular decomposition, used in [4], applies a grid over the AOI that results in square cells whose size allows them to be fully covered by the search vehicle once it enters them, removing the need for back-and-forth movements. The coverage is considered complete once all the cells are visited, but since a discretization is applied, this means that the AOI might either not be fully covered or some areas outside of it might be visited, depending on the cell selection criteria - hence an approximate decomposition. This is the method used in this work, with the implementation details specified in Section III. Finally, a no decomposition method can be applied [5], but these methods are unable to deal with complex-shaped AOIs and the presence of obstacles inside them, making them unsuitable for our application.

Some authors explore the use of prior information in surveillance missions, such as in [6], where target density and the probability of target detection in different types of terrain are taken into account by a Particle Swarm Optimization algorithm that generates the path of the search UAVs to maximize the number of targets detected. The use of prior information in maritime SAR scenarios is also addressed in [7], where the authors calculate a POC map by taking into account maritime currents and winds and use a dynamic programming approach to generate the path for a single UAV. This is the type of map used in this work to represent the available prior information. In [8], the authors estimate the risk of an area during a forest fire given its population density and accessibility, and introduce an algorithm that prioritizes searching high-risk areas first. Although these approaches succeed in exploiting the target-related information available, they do not consider the endurance constraints of the search vehicles.

One major obstacle in using small electrical UAVs for area coverage operations is their limited endurance. Studies such as [5] focus on minimizing energy consumption by examining the energy consumption profile of a small UAV across different flight phases and proposing an energy-aware CPP method. Inspired by this, [9] extends this approach to the multi-UAV case, optimizing the division of the AOI and planning the individual paths of the UAVs. In contrast, [10] develops an energy model for a quadcopter moving in a regular grid to estimate the energy consumption when applying approximate cellular decomposition to the AOI, which is the type of movement considered in this work. These algorithms, in turn, do not take advantage of any available prior information.

The advantages of using multiple UAVs over a single vehicle has been identified by several authors [4], [6], [8], [9], who extend their approaches to use multiple UAVs. This compensates for the limited flight time of small UAVs and enhances coverage speed by allowing simultaneous operation.

III. Problem Formulation

In this paper the goal is to produce a multi-UAV path planning algorithm that leverages available prior target position information and takes into account the limited endurance of small UAVs. The algorithm should also take advantage of the fact that the UAVs are air-dropped from a larger platform capable of deploying them at any position inside the AOI.

The Area of Interest (AOI) and No-fly Zones (NFZs) are defined as polygons on a two-dimensional plane, representing the regions UAVs need to search and avoid, respectively. We employ approximate cellular decomposition for the AOI, enhancing path optimization flexibility, detailing the Probability of Containment (POC) map, and simplifying coverage tracking. The cell size d in the grid used for the decomposition must ensure that a cell is completely covered by each UAV's sensor footprint. It is calculated as:

$$d = 2(1 - p_o)h \tan\left(\frac{FOV}{2}\right) \tag{1}$$

where p_o is the image overlap percentage, h is the flight altitude of the UAVs and FOV is the sensor's angular field of view. The sensor's target detection limitations should be considered when defining the altitude h. For instance, with $FOV = 84\,\text{deg}$, $h = 50$ m and $p_o = 50\%$, the grid size d is 114.6 m, as shown in Fig. 1.

A cell is valid if its center is within the AOI and outside any NFZs. UAVs can only move between centers of valid adjacent cells in the 8-connected neighborhood. The air-dropped UAVs can start from any valid cell, and the algorithm generates paths as sequences of cell center coordinates, as shown in Fig. 1.

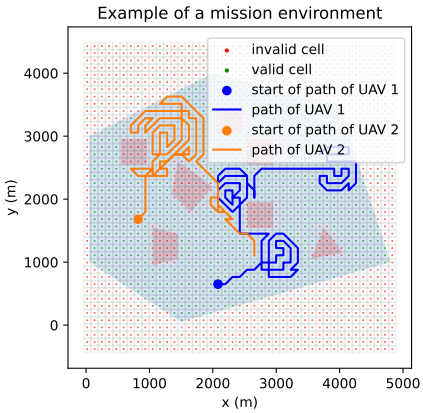


Fig. 1: Example of an area of operations - the blue region denotes the AOI, while the red regions indicate the NFZs. The paths of 2 UAVs obtained using the Attraction algorithm [8] are also presented.

Prior information about the potential target location must also be considered, which in SAR scenarios can be integrated as a POC map. Each valid cell has an associated probability indicating the likelihood of the target being contained within it, and the total sum of the POC values for all the valid cells equals 100%. An example of a POC map is shown in Fig. 2.

The output of the path planning algorithm is a set of paths X that contains the paths of the individual UAVs. The path of the UAV j is represented by X_j and consists of a sequence of adjacent cells. Therefore, $X_j(i)$ represents the i^{th} cell

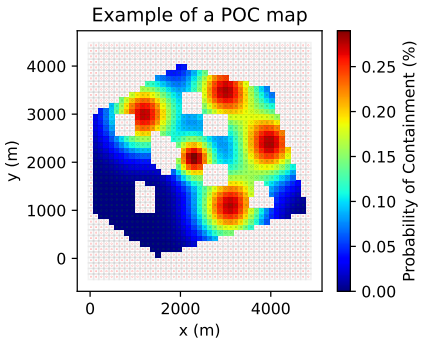


Fig. 2: Example of the prior information available in a SAR scenario - the color of each cell represents the probability that it contains the target. Only valid cells have an associated POC value. For this mission there are several possible target positions with different degrees of associated uncertainty.

visited by the UAV j. The energy limitations of the UAVs are considered using the energy model presented in [10]:

$$E(X_j) = 0.1164\ L(X_j) + 0.0173\ \theta(X_j) \tag{2}$$

where $E(X_j)$ is the energy consumption of the path X_j, $L(X_j)$ is the total length of the path X_j in meters and $\theta(X_j)$ is the sum of all the turn angles of the path X_j in degrees.[1] Given the maximum energy capacities of each of the UAVs, the algorithm must ensure that the energy demands of the path generated for each vehicle do not exceed its capacity.

In this study it is assumed that there is no advantage in revisiting a cell where no target was found. The set V_s of the cells visited before step s is defined as in:

$$V_s = \bigcup_{j=1}^{\mathrm{N_{UAV}}} \bigcup_{i=1}^{s-1} X_j(i) \tag{3}$$

where $\mathrm{N_{UAV}}$ is the total number of UAVs used. Let $I_{V_s}(x)$ be the indicator function of the set V_s, which has the value of 1 if the cell x has been visited before step s by any UAV and 0 otherwise, as in:

$$I_{V_s}(x) = \begin{cases} 1 & \text{if } x \in V_s, \\ 0 & \text{if } x \notin V_s \end{cases} \tag{4}$$

The goals in a SAR mission are to find the target and do so as fast as possible. Therefore, we can define two sub-objectives: maximizing the probability of detecting a target D, and minimizing the expected step in the path of the UAVs at which a target is detected EDS. The probability of target detection D given a set of UAV paths X can be calculated by:

$$D(X) = \sum_s \sum_{j=1}^{\mathrm{N_{UAV}}} \mathrm{POC}(X_j(s))\,[1 - I_{V_s}(X_j(s))] \tag{5}$$

[1]The model presented in [10] is associated with a small quadcopter and the energy unit used is kJ. However, since this work is platform-agnostic, arbitrary energy units are used.

where $\mathrm{POC}(X_j(s))$ is the POC value of the cell visited by UAV j in the step s. The term $[1 - I_{V_s}(X_j(s))]$ will take the value of 0 if the cell $X_j(s)$ has already been visited in the past by any UAV and 1 otherwise, thus ensuring that the POC of a certain cell is not considered twice in the overall sum.[2]

To calculate EDS, we first define the probability that the target is found at step s, given that the target is found by a set of UAV paths X:

$$\mathbb{P}(S = s|X) = \frac{\sum_{j=1}^{\mathrm{N_{UAV}}} \mathrm{POC}(X_j(s))\,[1 - I_{V_s}(X_j(s))]}{D(X)} \tag{6}$$

This consists of the sum of the POC of the cells visited by all UAVs for the first time at step s divided by the probability of target detection $D(X)$. Therefore, the EDS can be calculated as the expected step S at which the target will be found, given that it is detected by a set of UAV paths X:

$$EDS(X) = \mathbb{E}(S|X) = \sum_s s\,\mathbb{P}(S = s|X) \tag{7}$$

Replacing (5) and (6) in (7), we get:

$$EDS(X) = \frac{\sum_s s \sum_{j=1}^{\mathrm{N_{UAV}}} \mathrm{POC}(X_j(s))\,[1 - I_{V_s}(X_j(s))]}{\sum_s \sum_{j=1}^{\mathrm{N_{UAV}}} \mathrm{POC}(X_j(s))\,[1 - I_{V_s}(X_j(s))]} \tag{8}$$

In order to perform the paths optimization, an objective function J is required that reflects both of the sub-objectives described previously for SAR missions:

$$J(X) = \sum_s \sum_{j=1}^{\mathrm{N_{UAV}}} e^{-\epsilon s}\,\mathrm{POC}(X_j(s))\,[1 - I_{V_s}(X_j(s))] \tag{9}$$

where ϵ is a decay factor responsible for rewarding paths that visit sooner the cells with a high POC value, since $e^{-\epsilon s}$ will have a greater value for a lower s.[3] This objective function also rewards paths that visit more cells with a high POC value and that avoid revisiting cells. In other words, maximizing J means that the target is more likely to be found and that it is detected sooner on average. A lower ϵ value places less importance on rapid target detection, as reflected by the expected detection step EDS (7), allowing the algorithms to optimize the path for enhancing the likelihood of target detection D (5), and vice-versa. Therefore, this parameter allows for the adjustment of the output of the algorithm to best suit the application.

IV. Proposed Methods

Several methods were tested to solve the optimization problem described. A* with an admissible heuristic performs well for problems where the objective is to go from point A to point B and where the euclidean distance to the target can be used as an heuristic. However, for the problem described, the target position is unknown and therefore, finding an admissible

[2]In the case that multiple UAVs visit the same cell in the same step, the POC of that cell is only considered once in the overall sum. This eventuality is not contemplated in the equations so as not to overcomplicate the mathematical formulation. This is valid in all equations with the term $[1 - I_{V_s}(X_j(s))]$.

[3]The $e^{-\epsilon s}$ factor is equivalent to the γ^s discount usually applied to rewards in dynamic programming in order to increase the value of rewards in the short-term, and in this case $\gamma = e^{-\epsilon}$.

heuristic is not as straightforward and calculating it is much more computationally expensive. We also attempted formulating the problem as a Mixed-Integer Nonlinear Programming (MINLP) problem and use an off-the-shelf optimization tool, but also ran into scalability issues. Algorithms such as Particle Swarm Optimization (PSO) were considered, but these are more suitable for continuous optimization problems, which is the case in [6], where the path generated for the UAVs is continuous. Among the algorithms tested, Simulated Annealing (SA) and Ant colony Optimization (ACO) stood out as producing the best results, coping well with the scalability issues encountered by the other methods. Therefore SA and ACO are proposed in this paper as a tool to maximize the objective function presented in section III given the mission constraints. If provided an infinitely long temperature decay, with infinitely small decay steps, SA guarantees an optimal solution (stochastic convergence). However, in practice, the algorithm is time-flexible, allowing the trade of computational time for performance if necessary [11]. In contrast, ACO does not present any guarantees of finding an optimal solution.

A. Simulated Annealing (SA)

The pseudocode for the Simulated Annealing (SA) algorithm proposed is shown in Algorithm 1. Besides mission-specific inputs such as the area of operations, number of UAVs and energy capacity, some algorithm-specific inputs are also necessary, specifically the choice of the algorithm used to generate an initial solution guess $\mathcal{G}$, the initial temperature T_{init}, the cooling factor α, the minimum temperature T_{min}, the length of the Markov chains[4] L_k, the number of threads $n_{threads}$, and the boolean $Sync$ that indicates if the threads synchronize their accepted solutions at the end of each iteration of the Markov chains. The synchronization process used here for the multi-threaded SA algorithm is the one proposed in [12]. In turn, the function RUNINTHREAD() runs a single-threaded SA algorithm similar to the one proposed in [11].

For the initial solution guess, two algorithms are used: a Random Walk (RW) algorithm, which randomly chooses cells that the UAVs can move to while respecting the mission constraints; and an Attraction (Att) algorithm similar to the one in [8] that was modified so that it respects the mission constraints. This Attraction algorithm works by creating a path step by step for each UAV. At each step, the selected UAV chooses the cell in the AOI that produces the largest attraction (a value proportional to a cell's POC and inversely proportional to the distance to the UAV), and plots a course towards it that respects the cell adjacency constraint. The UAV then takes the first step in said course and repeats the process until all available energy runs out. This is contrary to the version proposed in [8], where the UAVs can move directly to the cell that produces the largest attraction, and the paths are only terminated once the entire area is searched.

Simulated Annealing compares an accepted solution with a neighboring candidate solution, and depending on the difference in performance, the candidate might become the accepted solution with a given probability. The function GENERATENEIGHBOUR(X_{accept}) is responsible for generating a neighbour solution to the one currently accepted. This is accomplished by randomly selecting one of the following modifications to perform:

- Remove a step in the path of a UAV, ensuring the new path does not violate any constraints and extend it randomly until all available energy is expended.
- Change a step in the path of a UAV and perform the same constraint and energy check as previously.
- Add a step in the path of a UAV. If the UAV's energy budget is exceeded, remove the excess steps.
- Remove an instance where a UAV path crosses itself or the path of another UAV. For a UAV crossing its own path, reverse the order of steps between the last step before the cross and the first step after. For two UAVs crossing, swap the rest of the path after the cross between them. An energy check is performed after the procedure.
- Change the initial position of the path of a UAV. A step in the path is randomly selected as the new starting position and the original path before it is deleted. The path is then extended until all the energy available is expended.

Algorithm 1 Multi-threaded Simulated Annealing algorithm

Inputs: $\mathcal{G}$, T_{init}, α, T_{min}, L_k, $n_{threads}$, $Sync$
Output: X ▷ A set of the paths of each UAV
$X_{init} \leftarrow$ initial guess of the set of paths generated with the algorithm chosen by $\mathcal{G}$
for $idx = 0 : n_{threads}$ **do**
  RUNINTHREAD(X_{init}, T_{init}, α, T_{min}, L_k, $Sync$)
end for
Wait for all threads to return a solution
$X \leftarrow$ the set of UAV paths that performs best out of all threads, i.e. has the lowest energy
return X

function RUNINTHREAD(X_{init}, T_{init}, α, T_{min}, L_k, $Sync$)
  $X_{accept} \leftarrow X_{init}$ ▷ The initial solution is accepted
  $E_{accept} \leftarrow -J(X_{accept})$ ▷ The minus sign is used since the goal is to maximize the objective function J as in (9) and SA minimizes the solution energy E
  $T \leftarrow T_{init}$
  while $T > T_{min}$ **do**
    for $l = 0 : L_k$ **do**
      $X_{cand} \leftarrow$ GENERATENEIGHBOR(X_{accept})
      $E_{cand} \leftarrow -J(X_{cand})$
      if random([0,1]) $< \exp\left(\frac{E_{accept} - E_{cand}}{T}\right)$ **then**
        $X_{accept} \leftarrow X_{cand}$ ▷ Accept the candidate solution
        $E_{accept} \leftarrow E_{cand}$
      end if
    end for
    if $Sync$ **then** ▷ Synchronizes the accepted solution
      Wait for all threads to reach this point
      $X_{accept} \leftarrow$ the accepted UAV paths (X_{accept}) with the lowest energy among all threads
    end if
    $T \leftarrow \alpha T$ ▷ Reduce the temperature
  end while
  return X_{accept} ▷ Thread returns its accepted solution
end function

B. Ant Colony Optimization (ACO)

The second algorithm proposed is based on Ant Colony Optimization (ACO), a type of approach explored by the authors in [13]. The pseudocode adapted for this application is described in Algorithm 2. Besides the problem-specific inputs,

[4]This parameter is not typical of SA, but is used in the implementation presented in [11], on which the algorithm presented was based.

the following algorithm-specific inputs are necessary: maximum number of iterations n_{iter}, influence of the pheromone value α, influence of the heuristic value β, pheromone evaporation rate ρ, pheromone deposit rate Q, and a minimum heuristic value η_{min}. We consider that the ants move in a decision tree, starting in a root node that allows the UAVs to access any valid cells in the AOI with a single step to simulate the flexible deployment position. To reduce the branching factor, we consider that only one UAV is allowed to move between consecutive nodes, alternating the move priority between the UAVs with sufficient energy to move to another cell.[5] This means that in each node transition, only one new cell in the AOI is visited, and its POC value affects the heuristic of that node.

In each iteration, an ant is placed on the root node, and until a terminal node is reached, the ant chooses a child node to move to according to the function CHOOSECHILD(). A node is considered to be terminal if all the cells in the AOI have been covered, or if none of the UAVs has enough energy remaining to move. If an ant reaches a node that has not been expanded yet, the function EXPAND() generates all the child nodes that can be reached from the current node and initializes them as indicated in NODESTRUCT(). This initialization process includes simulating the pheromone evaporation phenomenon that would have taken place if that node had existed since the beginning of the tree construction. After n_{iter} iterations, a special ant is placed in the root node and until it reaches a terminal node, it will choose the child node with the highest pheromone value. This special ant's path in the tree is then converted to a set of UAV paths by GETUAVPATHS() and returned by the algorithm.

V. EXPERIMENTS AND RESULTS

In sections V-B and V-C, several experiments are conducted in order to tune the parameters of both the SA and ACO algorithm, respectively. The resulting parameters are subsequently used in sections V-D and V-E. Section V-D shows the effect of the decay factor on the mission priorities, and in section V-E a comparison between the different algorithms is presented.

A. Test environment

A comprehensive test environment such as the one in Fig. 1 was used for the parameter tuning, with the POC map displayed in Fig. 2. This mission was performed by two UAVs, each with 2000 units of available energy. Fig. 1 also shows the paths resulting from applying the Attraction algorithm [8], adapted for our problem constraints and with random initial positions, since this algorithm is not able to optimize for the deployment positions.

[5]In a mission with n UAVs available and an AOI with m valid cells, if all UAVs are free to move when a node is expanded, in the root node expansion this would lead to m^n child nodes, and 8^n child nodes for any subsequent expansion. However, with the formulation used, only m child nodes are created in the expansion of nodes with a depth lower than n, and after that only 8 child nodes are created with each expansion. This allows the algorithm to focus on expanding the more promising nodes without having to generate all the possible deployment configurations.

Algorithm 2 Ant Colony Optimization (ACO) algorithm

Inputs: n_{iter}, α, β, ρ, Q, η_{min}
Output: X ▷ A set of the paths of each UAV
$root\_node \leftarrow$ NODESTRUCT(0) ▷ Initialize the root node, representing the system state before UAV deployment
for $iter = 0 : n_{iter}$ **do**
  $current\_node \leftarrow root\_node$
  $node\_path \leftarrow$ list[$current\_node$] ▷ Initialize a list with only the root node
  while $current\_node$ is not terminal **do**
    if $current\_node$ has not been expanded **then**
      EXPAND($current\_node$, $iter$)
    end if
    $current\_node \leftarrow$ CHOOSECHILD($current\_node$)
    Add $current\_node$ to $node\_path$
  end while
  for $node$ **in** all known nodes **do**
    $node.\tau \leftarrow (1-\rho)\, node.\tau$ ▷ Evaporate pheromones
  end for
  $J \leftarrow$ OBJECTIVEFUNCTION($node\_path$)
  for $node$ **in** $node\_path$ **do**
    $node.\tau \leftarrow node.\tau + QJ$ ▷ Deposit pheromones
  end for
end for
$final\_node\_path$ ←starting from the root node, chose the child with most pheromones until a terminal node is reached
return GETUAVPATHS($final\_node\_path$) ▷ Convert the node path to UAV paths

struct NODESTRUCT($iter$) ▷ Node structure initialized as follows at iteration $iter$
  $positions \leftarrow$ None ▷ Current positions of the UAVs
  $\tau \leftarrow (1-\rho)^{iter}$ ▷ Current pheromones (adjusted for evaporation)
  $children \leftarrow$ list[ ] ▷ List of child nodes initialized as empty list
end struct

function EXPAND($node$, $iter$) ▷ Generate the child nodes of the current node
  for $move$ in POSSIBLEMOVES($node$) **do**
    $child \leftarrow$ NODESTRUCT($iter$)
    $child.positions \leftarrow$ NEWPOSITIONS($node$, $move$)
    Add $child$ to $node.children$
  end for
end function

function CHOOSECHILD($node$) ▷ Choose a child node to move to
  $weights\_list \leftarrow$ [ ]
  for $child$ in $node.children$ **do**
    $new\_cell \leftarrow$ the cell to which the selected UAV moves, according to $child$
    if new_cell has been visited by one of the UAVs previously **then**
      $\eta \leftarrow \eta_{min}$
    else
      $\eta \leftarrow \eta_{min} +$ POC(new_cell)
    end if
    $weight \leftarrow child.\tau^{\alpha} \times \eta^{\beta}$ ▷ Ant decision formula
    Add $weigth$ to $weights\_list$ ▷ The larger the weight, the more likely a child node is to be chosen
  end for
  return A $child$ chosen randomly according to the weights in $weights\_list$
end function

B. Parameter tests for the Simulated Annealing algorithm

The performance of the SA algorithm depends on the choice of the algorithm-specific parameters described in section IV-A. Several tests were conducted to find the values for these parameters that maximize the performance. Due to the random nature of SA, the paths generated for a given mission are not always the same. Therefore, for each set of parameters tested, the algorithm was run 5 times. The average of the objective function (left vertical axis) is presented in a solid blue line, with the deviations from this average shown as a light-blue area, whereas the average computational time[6] is shown in a solid red line (right vertical axis). This testing methodology and result presentation also applies for the tests in section V-C.

[6]Processor: Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz, Number of cores: 24, RAM: 215 Gb

To determine the best values for the initial temperature T_{init}, several tests were conducted. Fig. 3 shows the result of varying the initial temperature when using SA with the Attraction algorithm (Att) as an initial path guess. The optimal value for T_{init} was therefore found to be 0.0004.

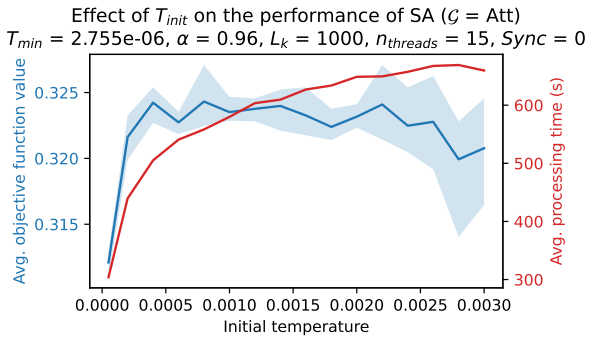


Fig. 3: Effect of the initial temperature T_{init} on the performance of SA. The average over 5 tests of the value of the objective function (left vertical axis) is plotted in a blue line and the average of computational time (right vertical axis) is represented in a red line.

Similar tests were performed to tune the other parameters and the results were the following:

- Tests for the cooling factor α showed that its increase leads to an exponential increase in processing time and that a value of 0.96 provides a good trade-off between performance and speed. However, depending on the time-frame available for the optimization, this parameter can be adjusted. Using 0.90 instead, reduces the time by half while maintaining an acceptable performance. The effect of this parameter is similar to that of L_k, for which the value of 1000 was used.
- It was concluded that if the number of threads $n_{threads}$ is increased, the resulting paths have a higher objective function value on average and the deviations from this average also decrease. A value of 15 was selected.
- The tests performed for T_{min} showed that this parameter does not significantly impact the performance and a value of 2.755×10^{-6} was found to yield good results.
- The synchronization of the threads was found to not result in a performance increase and was therefore not used.

C. Parameter tests for the Ant Colony Optimization algorithm

The optimization of the parameters for the ACO algorithm was done in a similar manner to the one in section V-B, with the intent of maximizing the algorithm's performance. Fig. 4 shows the effect of the number of iterations n_{iter} on the performance of the ACO algorithm. It was concluded that using a value greater than 20000 for this parameter did not translate to an improvement in performance and therefore this was the value used.

Contrary to the SA algorithm, where the optimal value for a parameter was independent from the others, the same cannot

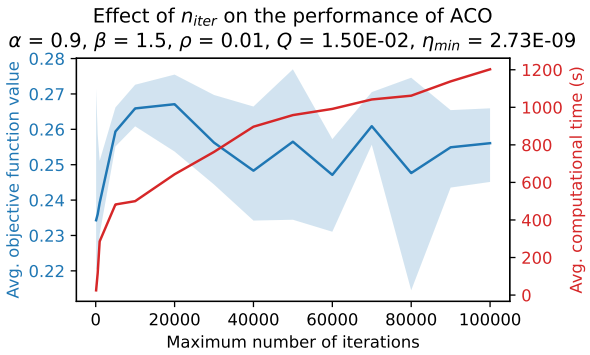


Fig. 4: Effect of the maximum number of iterations n_{iter} on the performance of ACO. The average over 5 tests of the value of the objective function (left vertical axis) is plotted in a blue line and the average of computational time (right vertical axis) is represented in a red line.

be said for ACO, since α, ρ and Q are all associated with the effect of the pheromones on the ant decision process, and β and η_{min} affect the heuristic component. Therefore, through a process of trial and error, the following values were found to yield good results: $\alpha = 0.9$, $\beta = 1.5$, $\rho = 0.01$, $Q = 0.015$. Since the heuristic chosen is the POC of the new cell that is visited when an ant chooses a certain node, the role of η_{min} is to reflect if that new cell has been visited in the past. This is necessary due to the fact that the heuristic cannot be zero, given the ant's decision function [13]. Therefore, η_{min} was chosen to be 0.1 times the minimum POC of the cells in the AOI. In this case, this means that $\eta_{min} = 2.73 \times 10^{-9}$.

D. Effect of the objective function's decay factor

The objective function, as outlined in section III, aims to represent the two goals of SAR missions: detecting the target and doing so as quickly as possible. To balance these objectives, a decay factor ϵ was introduced into the objective function J (9). Using the algorithm-specific parameters optimized in sections V-B and V-C, several tests were conducted for both the SA and ACO algorithms for different values of ϵ, with the goal of evaluating the impact of this parameter on the final result regarding the trade-off between the two sub-objectives. The results are presented in Fig. 5. As a baseline, the Boustrophedon algorithm was used,[7] and the result presented in the figure corresponds to the average performance over 100 random deployment positions. The Attraction algorithm [8] was also evaluated in a similar fashion. This randomization process is necessary since these algorithms do not optimize for the deployment position. Each of the remaining dots represents the average D (on the x axis) and EDS (on the y axis) over 5 tests for a given value of ϵ using

[7]This algorithm performs back-and-forth motions inside the AOI without taking into account the information available, as well as the energy constraints of the UAVs [2]. This is the algorithm typically used in SAR missions. In this case, the path is terminated when there is no remaining energy available.

the algorithm specified in the label. A Pareto front is also calculated for all the results and shown as a purple line.

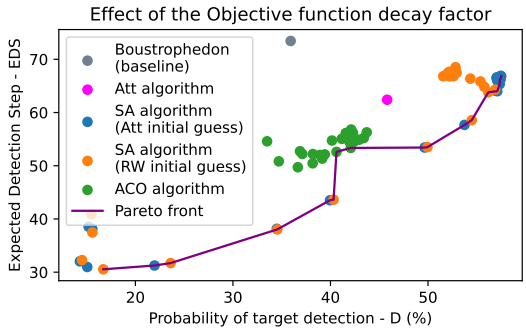


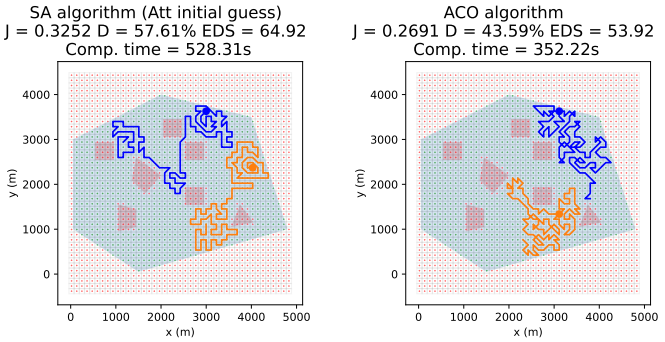
Fig. 5: Effect of the decay factor in the performance of the SA and ACO algorithms regarding the sub-objectives defined in (5) and (7). The dots in the top-right corner represent paths optimized for a low value of ϵ, whereas the ones in the bottom-left corner represent paths optimized for a high ϵ value.

The fact that the Pareto front contains mostly results from the SA algorithm indicates that this algorithm shows superior performance to the Boustrophedon, Attraction and ACO algorithms. It is also possible to note that SA can produce paths with very different trade-off characteristics regarding the sub-objectives as a result of changing ϵ. This is due to the fact that SA relies on comparing the performance between paths and a change in the metric that evaluates this (in this case the objective function J) will affect both of the paths that are being compared, meaning the algorithm will show good performance with a single set of parameters for a wide range of ϵ values. However, the same is not true for the ACO algorithm, for which the optimal algorithm-specific parameters heavily depend on the value of the objective function, which is influenced by ϵ. This means that this algorithm requires tuning of its parameters for each ϵ value used, which was not performed in the results in Fig 5. For all the tests presented in the previous sections, including for the parameter tuning, a value of 0.01 was used for ϵ, since it resulted in an acceptable trade-off between the two sub-objectives.

E. Algorithm results

In this section, the results of the algorithms discussed are presented, displaying the paths generated in different mission scenarios. For all the following tests, the algorithm-specific parameters tuned in sections V-B and V-C are used, as well as a decay factor of $\epsilon = 0.01$.

Fig. 6a shows the paths resulting from applying the SA algorithm for a mission scenario with the POC distribution of Fig.2, with 2 UAVs available, each with 2000 units of energy capacity. In turn, Fig. 6b shows the result of applying the ACO algorithm to the same mission scenario. In both cases, the algorithms place the deployment points of the UAVs in areas with a high density of cells that have high POC values in order to reduce the EDS.



(a) Paths resulting from applying the SA algorithm. All the major points of interest with high POC values were visited (see Fig. 2).

(b) Paths resulting from applying the ACO algorithm. The UAVs focus on the areas with high POC values.

Fig. 6: Examples of paths calculated for the mission scenario presented in Fig. 1 and Fig. 2, with 2 UAVs available, each with 2000 energy units.

To assess the performance of the SA algorithm with a larger number of UAVs, a test was conducted with 6 UAVs, each with 2000 energy units (Fig. 7). Compared to the previous mission scenario, the probability of target detection is increased, along with a decrease in the expected detection step. The effect of the number of available UAVs is illustrated in Fig. 8, where the SA algorithm was run for the same mission scenario with different numbers of UAVs. The mission success increases with the number of UAVs, with an increase in probability of target detection and a decrease in the expected target detection step. For more than 7 UAVs (2000 energy units each), the only improvement noted was the decrease of the EDS, with the detection probability remaining constant.

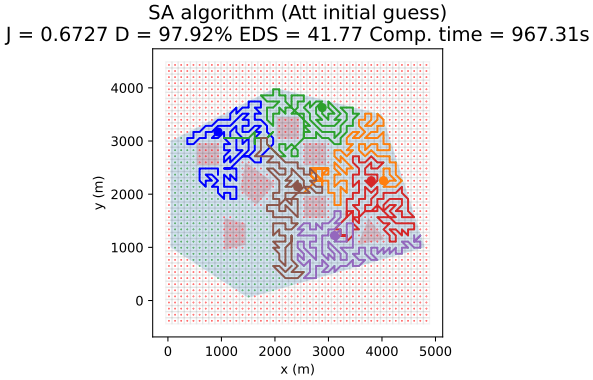


Fig. 7: Paths resulting from applying the SA algorithm to a mission scenario with 6 UAVs available, each with 2000 energy units. The UAVs efficiently cover the area, avoiding revisiting cells or crossing the paths of other UAVs. The search is also focused on the areas with high POC values, where the algorithm places the UAVs deployment positions.

A comparison of the performance of the different algorithms for this mission scenario is presented in Table I. The SA

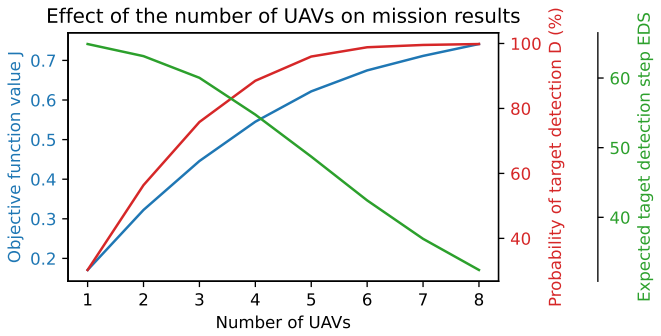


Fig. 8: Effect of the number of UAVs in the mission success. Results obtained with the SA algorithm. The three performance metrics are presented in three vertical axes: J in blue, D in red and EDS in green.

algorithm results in the best performance overall in terms of the objective function J and presents an acceptable increase in computational time when N_{UAV} increases. Another advantage of the SA algorithm is that it allows the reduction of the computational time at the cost of a worse (but still acceptable) performance, as can be seen for $\alpha = 0.90$. On the other hand, the computational time of the ACO algorithm exponentially increases with N_{UAV}, and its performance is on par with the Attraction algorithm that requires less computational time. Nonetheless, both of the approaches proposed perform better than the Boustrophedon baseline.

TABLE I: A mission scenario was tested for different numbers of available UAVs, $\mathbf{N_{UAV}}$. The column **Algorithm** displays the methods employed, and the rest of the columns show their average performance.

N_{UAV}	Algorithm	J	D (%)	EDS	Comp. time[6] (s)
2	Boustrophedon	0.1903	36.28	73.62	**0.22**
	Attraction	0.2656	46.17	62.27	0.41
	SA ($\alpha = 0.96$)	**0.3252**	**57.61**	64.92	528.31
	SA ($\alpha = 0.90$)	0.3203	56.30	63.90	213.98
	ACO	0.2691	43.59	**53.92**	352.22
6	Boustrophedon	0.4675	87.67	67.48	**0.71**
	Attraction	0.5950	92.66	49.74	1.10
	SA ($\alpha = 0.96$)	**0.6727**	97.92	**41.77**	967.31
	SA ($\alpha = 0.90$)	0.6685	**98.14**	43.01	417.17
	ACO	0.5795	85.03	42.70	5351.74

VI. Conclusions and Future Work

In this work, a multi-UAV path planning algorithm was proposed for SAR missions with the goal of finding the target as soon as possible. The algorithms proposed take into account prior information in the form of a POC map and the energy limitations of the search vehicles, leveraging the availability of multiple UAVs, as well as their flexible deployment positions.

Firstly, a decomposition method for the AOI was implemented, considering the mission requirements and the available UAV systems. Secondly, an objective function was formulated, that represents the goals of a SAR mission, balancing the probability of target detection and the time required to find it, with a parameter to adjust the desired trade-off between these objectives. Subsequently, a Simulated Annealing (SA) and an Ant Colony Optimization (ACO) algorithm were proposed, the parameters for which were tuned using a comprehensive test scenario. Finally, their performance was compared against a baseline Boustrophedon algorithm and the impact of using multiple UAVs for SAR missions was studied.

Future research could further optimize the parameters for the ACO algorithm, potentially using Bayesian optimization [14]. The use of neural MCTS approaches might also contribute to a more time-efficient optimization process for the UAV paths, given the similarity in search space complexity to the game of "Go", where a time-efficient solution was proposed with this type of method in [15].

References

[1] J. Bento, M. Basiri, and R. Ventura, "Information-Oriented and Energy-Aware Path Planning for Small Unmanned Aerial Vehicles," in *Progress in Artificial Intelligence: 23rd EPIA Conference on Artificial Intelligence*. Cham: Springer Nature Switzerland, September 2024, in press.

[2] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001. [Online]. Available: https://doi.org/10.1023/A:1016639210559

[3] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," in *Field and Service Robotics*, G. Ishigami and K. Yoshida, Eds. Singapore: Springer Singapore, 2021, pp. 277–290.

[4] A. Kapoutsis, S. Chatzichristofis, and E. Kosmatopoulos, "Darp: Divide areas algorithm for optimal multi-robot coverage path planning," *Journal of Intelligent & Robotic Systems*, vol. 86, 06 2017.

[5] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in *Proceedings - 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2015*, 04 2015.

[6] R. R. Pitre, X. R. Li, and R. Delbalzo, "Uav route planning for joint search and track missions—an information-value approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2551–2565, 2012.

[7] J. Ren, K. Liu, Y. Cui, and W. Du, "Search path planning algorithm based on the probability of containment model," *Mathematical Problems in Engineering*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:234058033

[8] V. San Juan, M. Santos Peñas, and J. Andujar Marquez, "Intelligent uav map generation and discrete path planning for search and rescue operations," *Complexity*, vol. 2018, pp. 1–17, 04 2018.

[9] S. Liu, X. Li, M. Meng, and X. Gong, "Energy-aware coverage path planning of multi-uav based on relative distance scaling cluster method," in *2023 International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2023, pp. 709–714.

[10] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "Ub-anc planner: Energy efficient coverage path planning with multiple drones," in *2017 IEEE international conference on robotics and automation (ICRA)*, 05 2017, pp. 6182–6189.

[11] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated annealing: From basics to applications," *Handbook of metaheuristics*, pp. 1–35, 2019.

[12] J. García-Rodríguez, C. Vázquez, J. G. López-Salas, and A. Ferreiro, "An efficient implementation of parallel simulated annealing algorithm on gpus," *Journal of Global Optimization*, vol. 57, 11 2012.

[13] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[14] E. Yin and K. Wijk, "Bayesian parameter tuning of the ant colony optimization algorithm: Applied to the asymmetric traveling salesman problem," 2021.

[15] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.